Relatório de CT-213: Detecção de objetos

Henrique F. Feitosa

Instituto Tecnológico de Aeronáutica,
São José dos Campos, São Paulo, Brasil

1 Introdução

Nessa prática, buscou-se implementar o algoritmo YOLO(*You Only Look Once*), que se utiliza de uma rede neural convolucional com a arquiterura descrita pela figura 1.

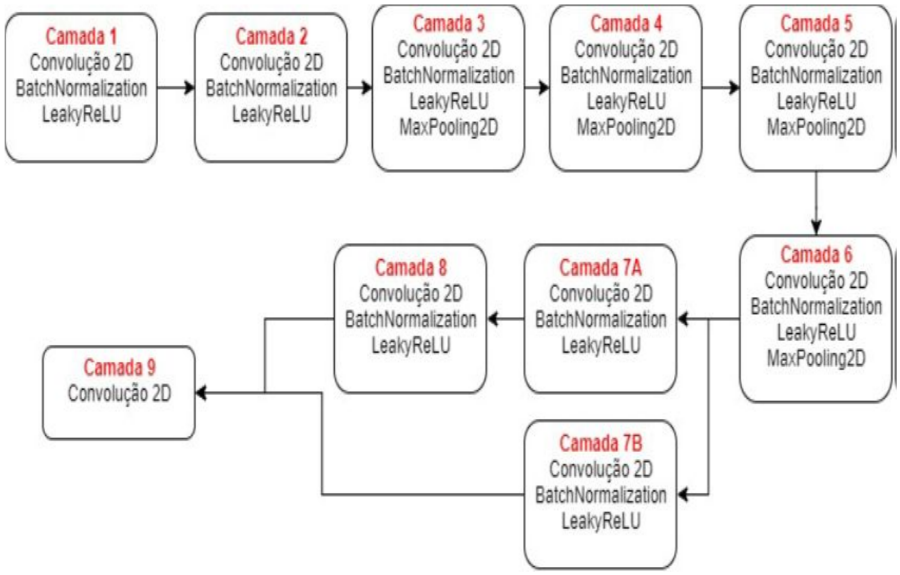

Figura 1. Arquitetura da rede neural LeNet-5.

Inicialmente, montou-se a rede neural usando o framework *keras* com backend *tensor flow*. Porém, como o treinamento demorava muito, não se realizou o treinamento da rede neural. Após isso, implementou-se o algoritmo YOLO e tentou-se identificar três objetos em uma série de imagens: uma bola e duas traves.

2 Resultados e Discussão

Para conferir que a implementação da rede estava igual a pedida, as figuras 2 e 3 mostra uma tabela que descreve a implementação da rede neural.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 120, 160, 3)	0	
conv_1 (Conv2D)	(None, 120, 160, 8)	216	input_1[0][0]
norm_1 (BatchNormalization)	(None, 120, 160, 8)	32	conv_1[0][0]
leaky_relu_1 (LeakyReLU)	(None, 120, 160, 8)	0	norm_1[0][0]
conv_2 (Conv2D)	(None, 120, 160, 8)	576	leaky_relu_1[0][0]
norm_2 (BatchNormalization)	(None, 120, 160, 8)	32	conv_2[0][0]
leaky_relu_2 (LeakyReLU)	(None, 120, 160, 8)	0	norm_2[0][0]
conv_3 (Conv2D)	(None, 120, 160, 16)	1152	leaky_relu_2[0][0]
norm_3 (BatchNormalization)	(None, 120, 160, 16)	64	conv_3[0][0]
leaky_relu_3 (LeakyReLU)	(None, 120, 160, 16)	0	norm_3[0][0]
max_pool_3 (MaxPooling2D)	(None, 60, 80, 16)	0	leaky_relu_3[0][0]
conv_4 (Conv2D)	(None, 60, 80, 32)	4608	max_pool_3[0][0]
norm_4 (BatchNormalization)	(None, 60, 80, 32)	128	conv_4[0][0]
leaky_relu_4 (LeakyReLU)	(None, 60, 80, 32)	0	norm_4[0][0]
max_pool_4 (MaxPooling2D)	(None, 30, 40, 32)	0	leaky_relu_4[0][0]
conv_5 (Conv2D)	(None, 30, 40, 64)	18432	max_pool_4[0][0]
norm_5 (BatchNormalization)	(None, 30, 40, 64)	256	conv_5[0][0]
leaky_relu_5 (LeakyReLU)	(None, 30, 40, 64)	0	norm_5[0][0]
max_pool_5 (MaxPooling2D)	(None, 15, 20, 64)	0	leaky_relu_5[0][0]
conv_6 (Conv2D)	(None, 15, 20, 64)	36864	max_pool_5[0][0]
norm_6 (BatchNormalization)	(None, 15, 20, 64)	256	conv_6[0][0]
leaky_relu_6 (LeakyReLU)	(None, 15, 20, 64)	0	norm_6[0][0]
max_pool_6 (MaxPooling2D)	(None, 15, 20, 64)	0	leaky_relu_6[0][0]
conv_7A (Conv2D)	(None, 15, 20, 128)	73728	max_pool_6[0][0]
norm_7A (BatchNormalization)	(None, 15, 20, 128)	512	conv_7A[0][0]
leaky_relu_7A (LeakyReLU)	(None, 15, 20, 128)	0	norm_7A[0][0]

Figura 2. Mostra a descrição da rede neural implementada.

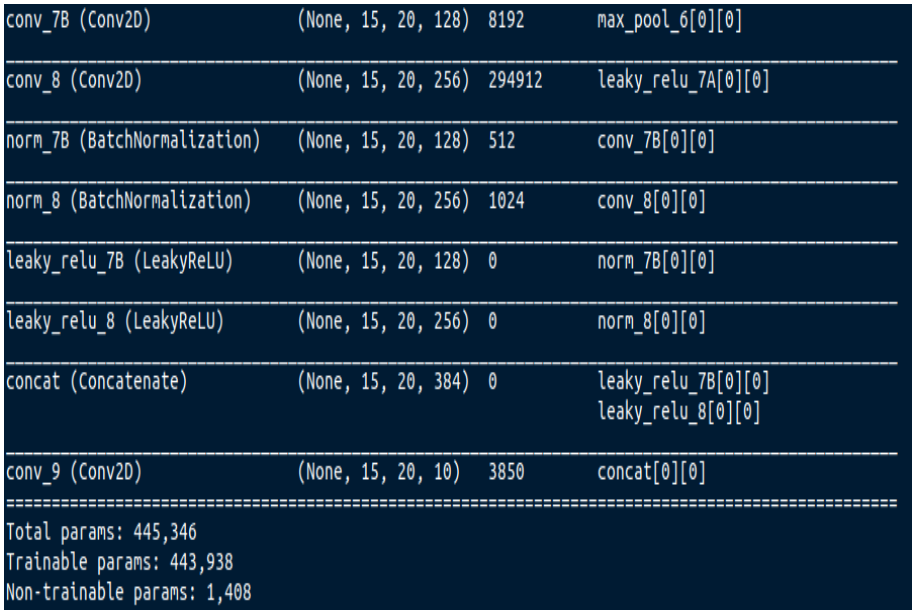

Figura 3. Mostra a descrição da rede neural implementada.

Assim, percebe-se que a rede neural implementada está de acordo com as exigências.

As figuras 4 a 8 mostram alguns resultados da detecção de objetos.

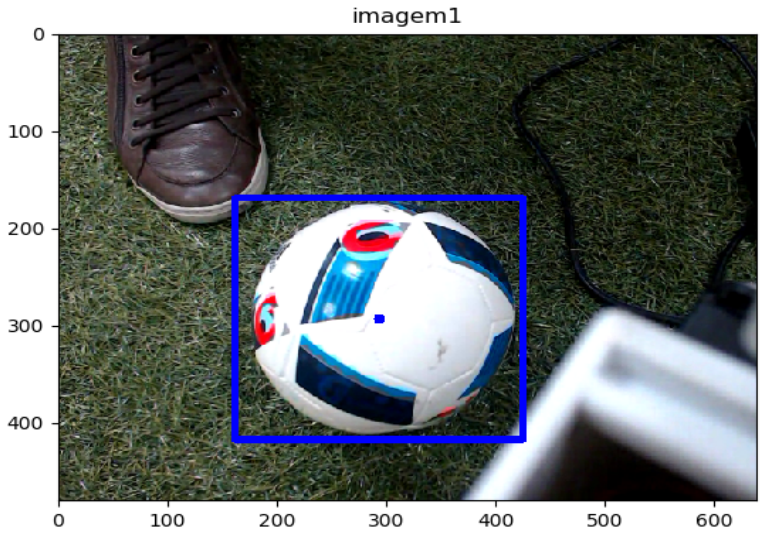

Figura 4. Mostra o resultado da detecção pelo algoritmo YOLO

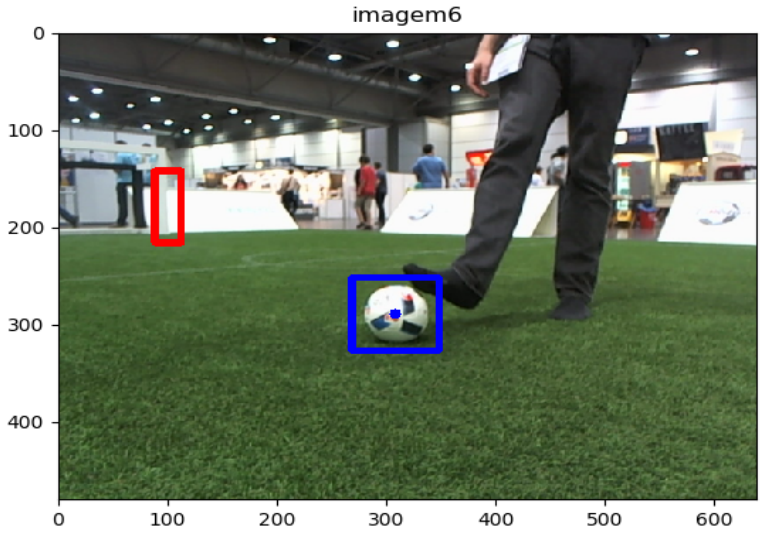

Figura 5. Mostra o resultado da detecção pelo algoritmo YOLO

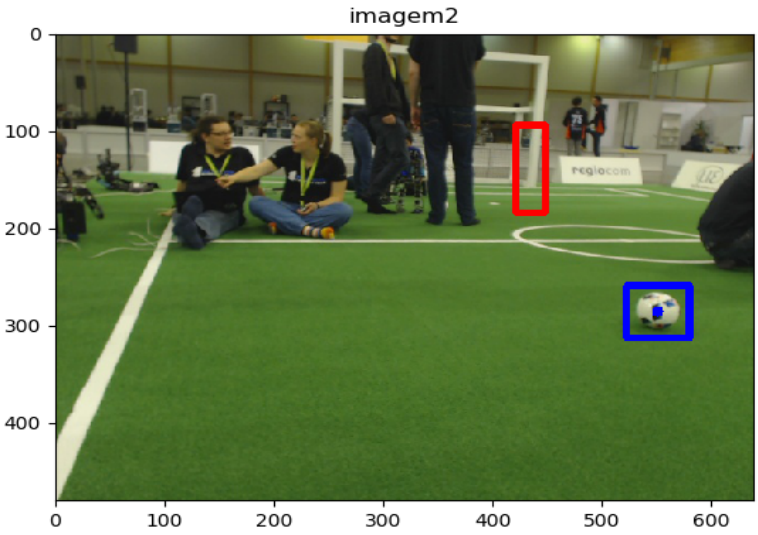

Figura 6. Mostra o resultado da detecção pelo algoritmo YOLO

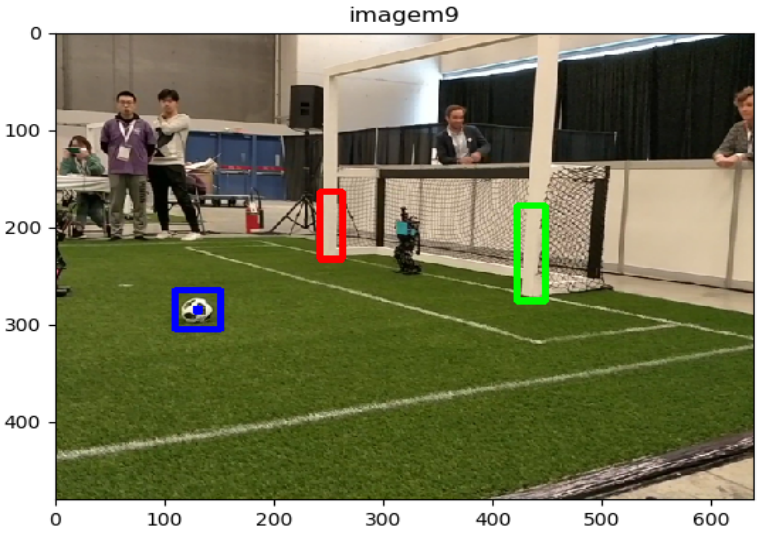

Figura 7. Mostra o resultado da detecção pelo algoritmo YOLO

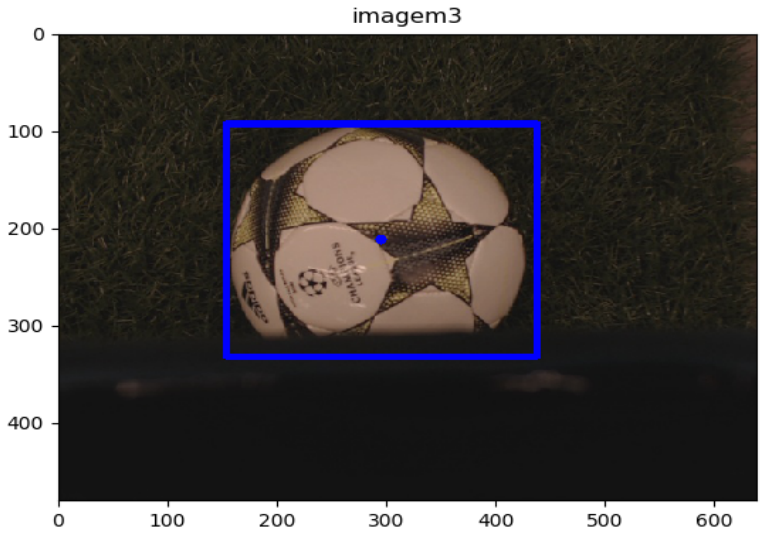

Figura 8. Mostra o resultado da detecção pelo algoritmo YOLO

Assim, percebe-se que o algoritmo tem desempenho satisfatório tanto quando os três objetos estaõ bem evidentes, como nas figuras 7 e 4, ou quando há obstáculos que dificultam a visão, como as figuras 8 e 5. Além disso, vale destacar alguns casos em que a identificação falha, como a figura 6, mas isso se deve ao número de elementos excessivos que estão se sobrepondo na imagem, o que pode ter causado uma confusão no processo de classificação da rede.